Example

EMPLOYEE2

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	SPSS	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

Question – Is this a relation?

Answer – Yes: unique rows and no multivalued attributes

Question – What's the primary key?

Answer – Composite: Emp_ID, Course_Title

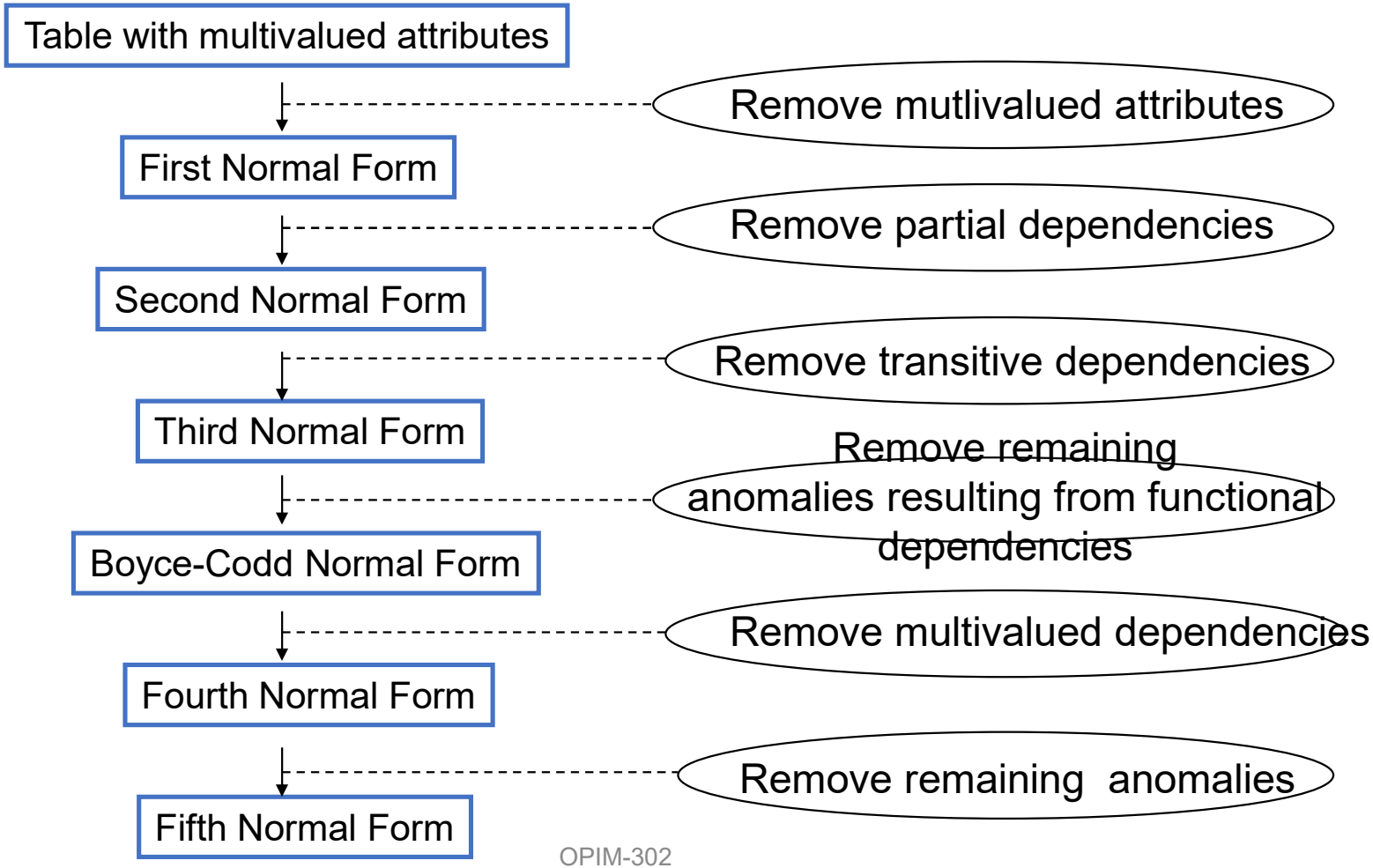
Anomalies in this Table

- **Insertion** – can't enter a new employee without having the employee take a class
- **Deletion** – if we remove employee 140, we lose information about the existence of a Tax Acc class
- **Modification** – giving a salary increase to employee 100 forces us to update multiple records

Why do these anomalies exist?

Because there are two themes (entity types) into one relation. This results in duplication, and an unnecessary dependency between the entities

Steps in Normalization



First Normal Form

- A relation that contains no repeating groups (multi valued attributes)
- Every attribute value is atomic

Removing Multi valued Attributes

- Simply fill in the relevant data values in vacant cells
- A new relation is created that has only single-valued attributes and satisfies rule 2

Removing Multi valued Attributes

Employee1 Relation

ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48000	SPSS	6/19/1999
				Surveys	10/7/1999
140	Alan Beeton	Accounting	52000	Tax Acc	12/8/1998
110	Chris Lucero	Info. Systems	43000	SPSS	1/12/1999
				C++	4/22/1999
190	Lorenzo Davis	Finance	55000		
150	Susan Martin	Marketing	42000	SPSS	6/16/1999
				Java	8/12/1999

Removing Multi valued Attributes

Employee2 Relation

ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48000	SPSS	6/19/1999
100	Margaret Simpson	Marketing	48000	Surveys	10/7/1999
140	Alan Beeton	Accounting	52000	Tax Acc	12/8/1998
110	Chris Lucero	Info. Systems	43000	SPSS	1/12/1999
110	Chris Lucero	Info. Systems	43000	C++	4/22/1999
190	Lorenzo Davis	Finance	55000		
150	Susan Martin	Marketing	42000	SPSS	6/16/1999
150	Susan Martin	Marketing	42000	Java	8/12/1999

Functional Dependencies and Keys

- Functional Dependency: The value of one attribute (the ***determinant***) determines the value of another attribute
- Candidate Key:
  - A unique identifier. One of the candidate keys will become the primary key
    - E.g. perhaps there is both credit card number and SS# in a table…in this case both are candidate keys
  - Each non-key field is functionally dependent on every candidate key

Functional Dependency

- An attribute B is functionally dependent on attribute A if for every valid instance of A, the value of A uniquely determines the value of B

Notation: A → B

EX: (EMPID, COURSE, DATE COMPLETED)

EMPID, COURSE → DATE COMPLETED

Ex: Functional Dependency

SSN → NAME, BIRTHDATE

VIN → MAKE, MODEL, COLOR

ISBN → TITLE

- Determinant: Attribute on the left-hand side of the arrow

Rules of Functional Dependency

Augmentation: If X→Y then XZ→Y

EX: Student# → Student Name then Student#, Course→Student Name

Transitivity: If X→Y and Y→Z then X→Z

EX: Student#→Major & Major→Advisor then Student#→Advisor

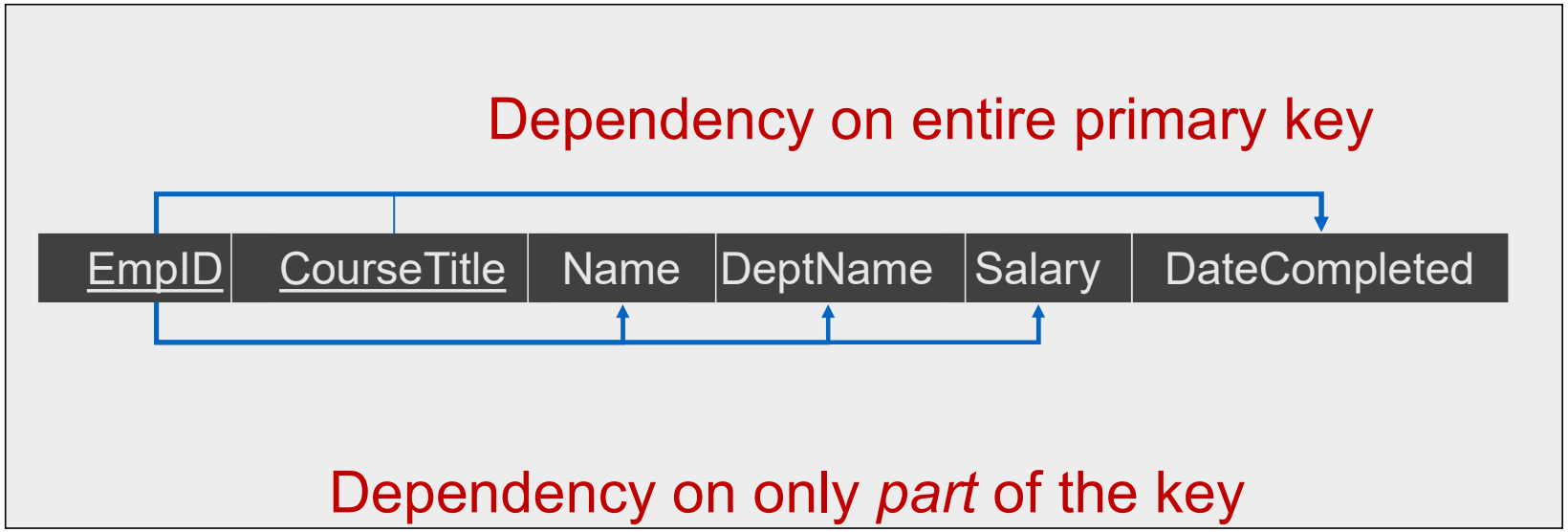
Second Normal Form

- A relation is 1NF and
  - *The Primary Key consists of only one attribute*
  - No Non-key attributes exist
  - Every non-key attribute is fully functionally dependent on the full set of primary key attributes, not by only part of the key
  - No partial functional dependencies

Functional dependency olarak primary key closure'nda ise

A -> B,C

Functional Dependencies in EMPLOYEE2



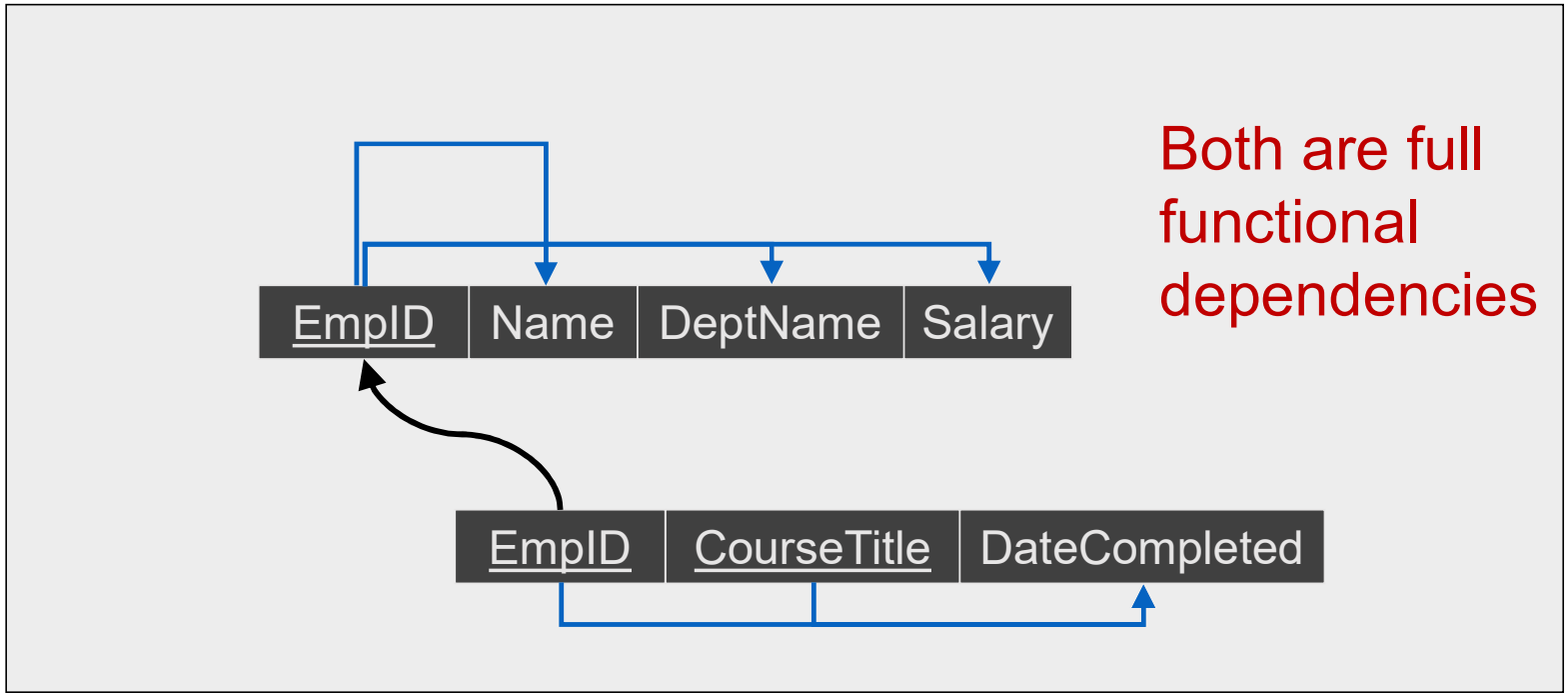
EmpID, CourseTitle ➔ DateCompleted

EmpID ➔ Name, DeptName, Salary

Therefore, NOT in 2nd Normal Form!!

Getting it into 2nd Normal Form

- Decompose into two separate relations



Third Normal Form

- 2NF PLUS ***no transitive dependencies*** (one attribute functionally determines a second, which functionally determines a third)
- Transitive dependency is the dependency between or among two non-key attributes

Removing Transitive Dependencies

- Decompose relation into two (or more) relations.
- The determinant of the transitive dependency becomes the primary key in the new relation and a foreign key in the original relation

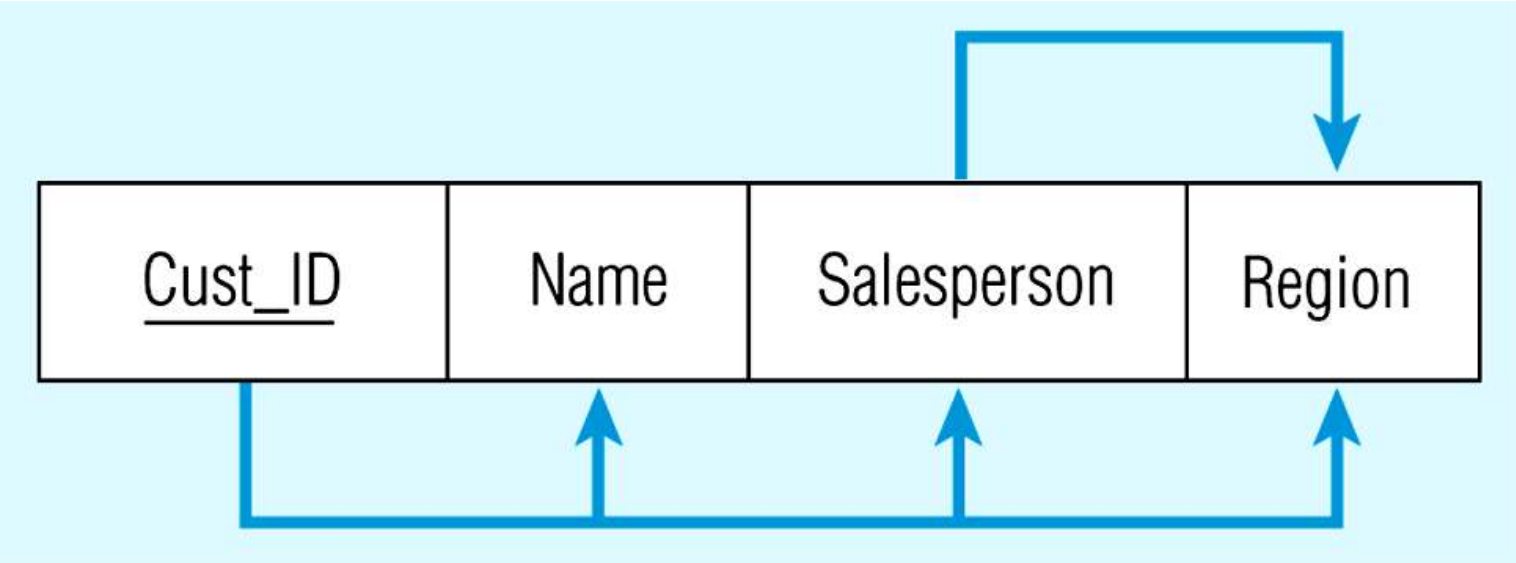
Relation with transitive dependency

(a) SALES relation with simple data

SALES

Cust_ID	Name	Salesperson	Region
8023	Anderson	Smith	South
9167	Bancroft	Hicks	West
7924	Hobbs	Smith	South
6837	Tucker	Hernandez	East
8596	Eckersley	Hicks	West
7018	Arnold	Faulb	North

Relation with transitive dependency



CustID → Name
CustID → Salesperson
CustID → Region

All this is OK
(2nd NF)

BUT

CustID → Salesperson → Region

Transitive dependency
(not 3rd NF)

Removing a transitive dependency

(a) Decomposing the SALES relation

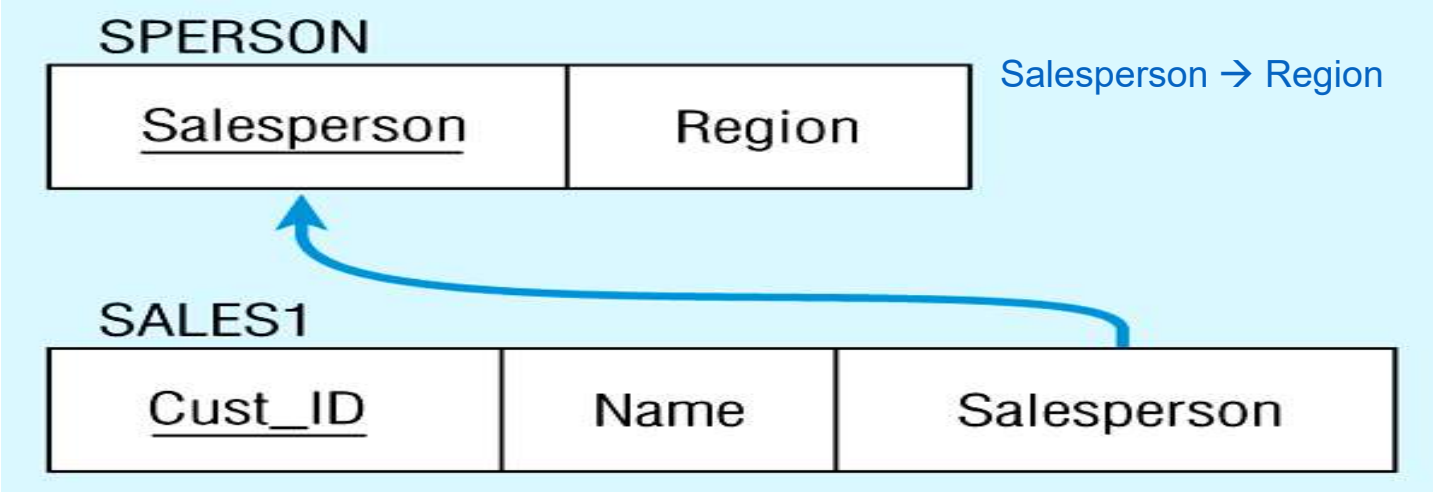
SALES1

Cust_ID	Name	Salesperson
8023	Anderson	Smith
9167	Bancroft	Hicks
7924	Hobbs	Smith
6837	Tucker	Hernandez
8596	Eckersley	Hicks
7018	Arnold	Faulb

SPERSON

Salesperson	Region
Smith	South
Hicks	West
Hernandez	East
Faulb	North

Relations in 3NF



CustID → Name

CustID → Salesperson

Now, there are no transitive dependencies…
Both relations are in 3rd NF

Ex: Transitive Dependency

- Relation:
  - Shipment(Snum, **Origin, Dest**, Distance)
- Dependencies:
  - Snum→Origin, Dest, Distance
  - Origin, Dest→Distance
- Fix by decomposing into two relations:
  - Shipto(Snum, Origin, Dest)
  - Distance(Origin, Dest, Distance)

Removing Multi valued Attributes

Employee2 Relation

ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48000	SPSS	6/19/1999
100	Margaret Simpson	Marketing	48000	Surveys	10/7/1999
140	Alan Beeton	Accounting	52000	Tax Acc	12/8/1998
110	Chris Lucero	Info. Systems	43000	SPSS	1/12/1999
110	Chris Lucero	Info. Systems	43000	C++	4/22/1999
190	Lorenzo Davis	Finance	55000		
150	Susan Martin	Marketing	42000	SPSS	6/16/1999
150	Susan Martin	Marketing	42000	Java	8/12/1999

ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48000	SPSS	6/19/1999
100	Margaret Simpson	Marketing	48000	Surveys	10/7/1999
140	Alan Beeton	Accounting	52000	Tax Acc	12/8/1998
110	Chris Lucero	Info. Systems	43000	SPSS	1/12/1999
110	Chris Lucero	Info. Systems	43000	C++	4/22/1999
190	Lorenzo Davis	Finance	55000		
150	Susan Martin	Marketing	42000	SPSS	6/16/1999
150	Susan Martin	Marketing	42000	Java	8/12/1999

1NF: it is already in 1NF. All entries are atomic.

2NF: fully functionally dependent upon the entire primary key.

ID → Name, Dept_Name, Salary

ID, Course_title → Date_Completed

[ID, Name, Dept_name, Salary]

[ID, Course_Title, Date_Completed]

Now onto the 3NF:
No transitive dependencies among non-key attributes.

Conclusion: no possible violation of 3nf.

So the set of tables are now in 3NF.

Merging Relations

- After transforming E-R diagrams into relations
  - Check each relation to ensure 3NF (at least)
  - Normalize if necessary
  - Check for redundancy - merge redundant relations

Merging Relations

- Issues to watch out for when merging entities from different ER models:
  - Synonyms – two or more attributes with different names but same meaning. E.g. Student ID & Matriculation#
  - Homonyms – attributes with same name but different meanings. E.g. Patient# could represent inpatient or outpatient

Merge Relations

- Transitive dependencies – even if relations are in 3NF prior to merging, they may not be after merging

  Student1(StudentID, Major)
  Student2(StudentID, Advisor)

  Student(StudentID, Major, Advisor)
  Major→Advisor

  Student(STDID, Major, Advisor)
  Major→Advisor

  Student(STDID, Major)
  Advisor(Major, Advisor)

- Supertype/subtype relationships – may be hidden prior to merging

  STUDENT [Student_ID, Major]

  ST_ADVISOR [Major, Advisor]

Ex: Merge Relations

Employee1(EmployeeNo, Name, Address, Phone)

Employee2(EmployeeNo, Name, Address, Jobcode, NoYrs)

Hint: Look for same primary key

Employee (EmployeeNo, Name, Address, Phone, Jobcode, NoYrs)

Integrity Constraints

- Domain Constraints: the set of values that may be assigned to an attribute must be from the same domain
  - Domain name, meaning, data type, size/length, allowable values or ranges
- Entity Integrity: Every relation must have a primary key and the data values of that key are valid.
  - RULE: The primary key or any component of the primary key can never be null

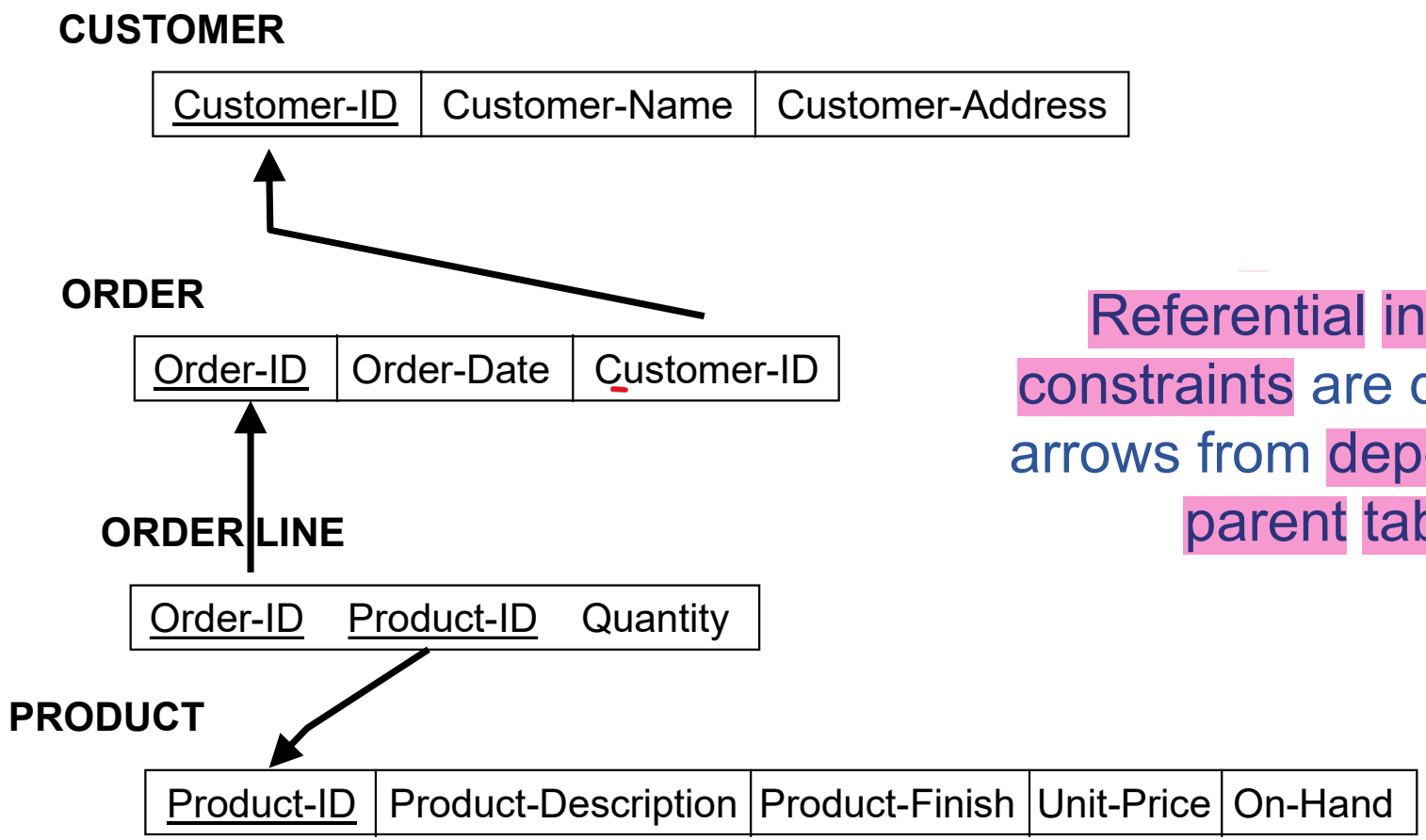
Integrity Constraints

- Referential Integrity – rule that states that any foreign key value (on the relation of the many side) MUST match a primary key value in the relation of the one side. (Or the foreign key can be null)

  Delete Rules

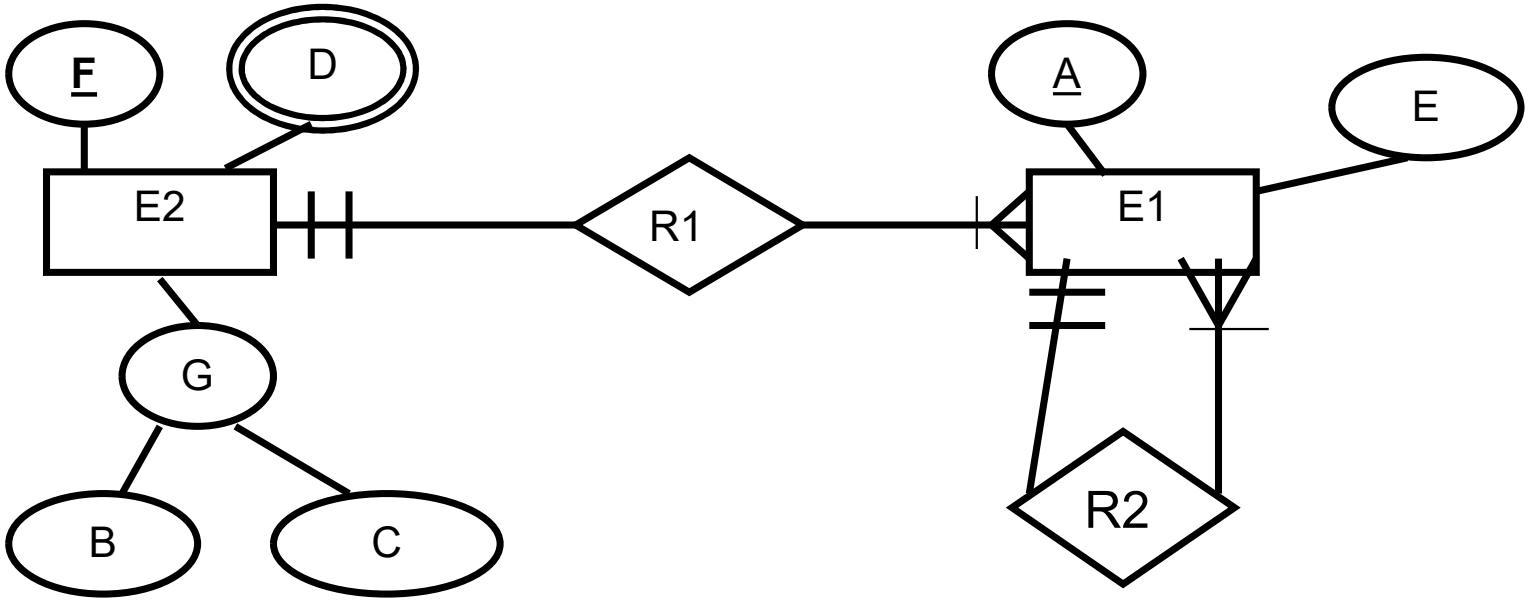
  - Restrict – don't allow delete of "parent" side if related rows exist in "dependent" side
  - Cascade – automatically delete "dependent" side rows that correspond with the "parent" side row to be deleted
  - Set-to-Null – set the foreign key in the dependent side to null if deleting from the parent side → not allowed for weak entities

Referential Integrity Constraints

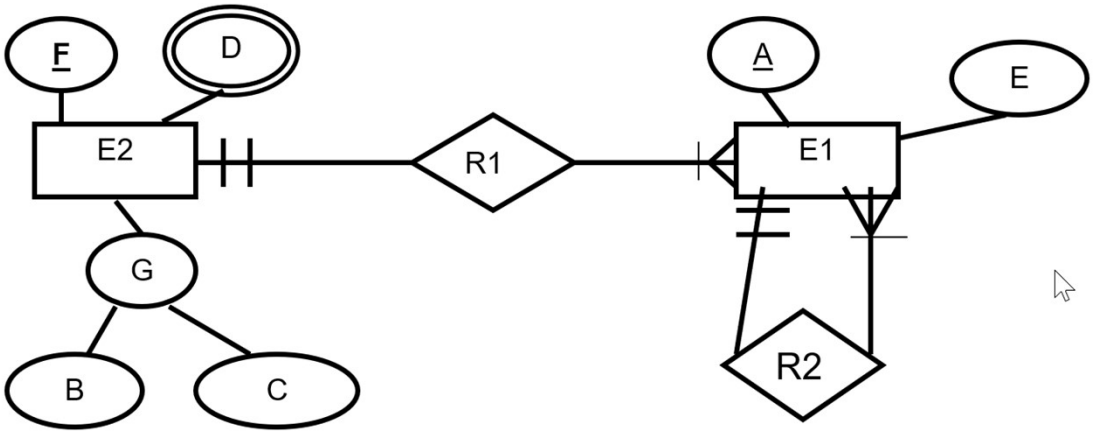


Referential integrity constraints are drawn via arrows from dependent to parent table

In Class Exercise 7



Transform the ER diagram given above on to logical data model



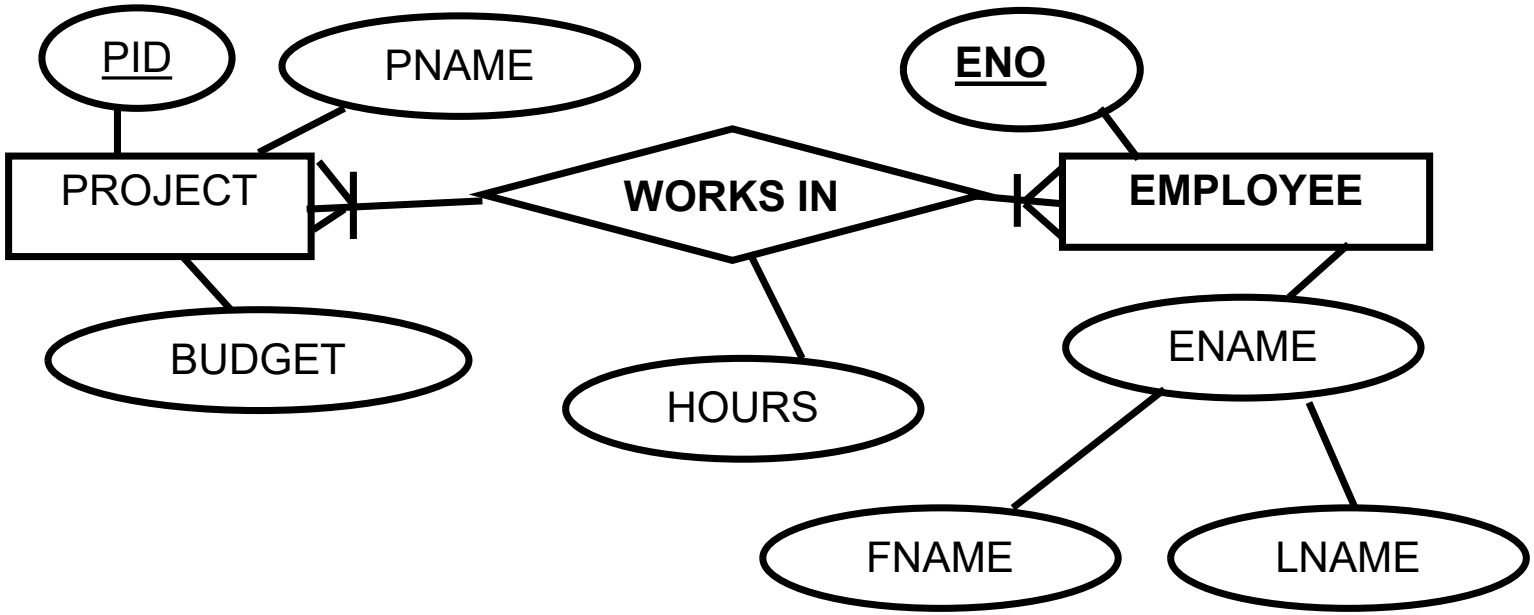
~~E2 [F , B, C, D] ← not 1st nf. Because of multivalued attrib.~~

E2_1[F, B, C]

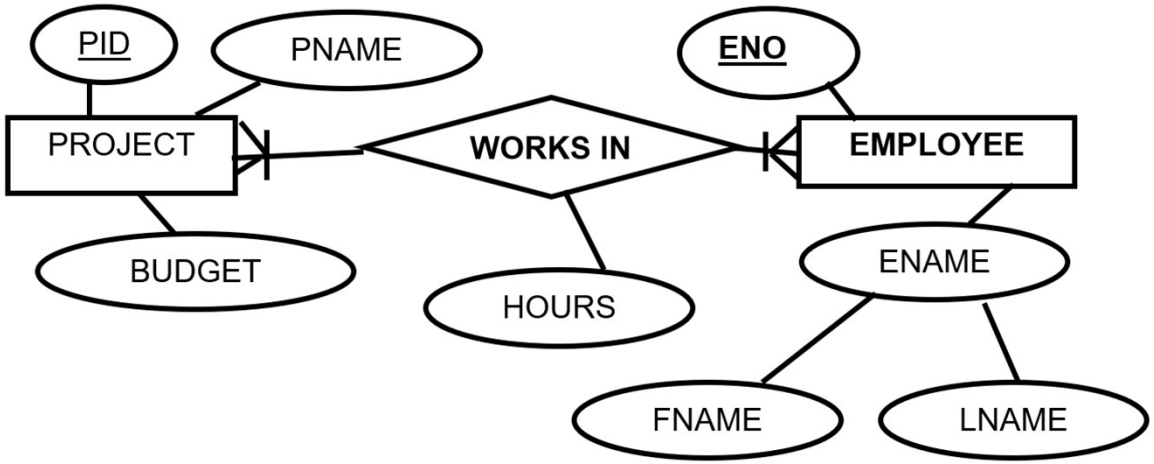
E2_2 [F, D]

E1 [A, E, F..A1]

In Class Exercise 8



Transform the ER diagram given above on to logical data model



PROJECT [PID, PNAME, BUDGET]

WORKS_IN [PID, ENO , HOURS]

EMPLOYEE [ENO, FNAME, LNAME]

In Class Exercise 9

Given dependencies below, normalize following relations and obtain 3NF relations.

PNO→ PNAME, BUDGET

ENO→ ENAME, TITLE, SALARY

ENO, PNO→ RESPONSIBILITY, DURATION

TITLE→ SALARY

EMPLOYEE

ENO	ENAME	TITLE	SALARY	PNO	RESPONSIBILITY	DURATION

PROJECT

PNO	PNAME	BUDGET

PNO→ PNAME, BUDGET

ENO→ ENAME, TITLE, SALARY

ENO, PNO→ RESPONSIBILITY, DURATION

TITLE→ SALARY

EMPLOYEE

ENO	ENAME	TITLE	SALARY	PNO	RESPONSIBILITY	DURATION

PROJECT

PNO	PNAME	BUDGET

EMPLOYEE [ENO, ENAME, TITLE, [redacted], PNO, [redacted]]

EMPLOYEE_PR [ENO, PNO, RESPONSIBILITY, DURATION]

EMP_TITLE [TITLE, SALARY]

PROJECT [PNO, PNAME, BUDGET]